线程池：

1. 任务队列 生产者-消费者模型

生产者线程向队列中添加新任务 消费者线程从队列中取出任务并执行

1. 工作的线程(消费者)

任务队列的任务都是函数的地址，消费者线程基于函数地址进行调用

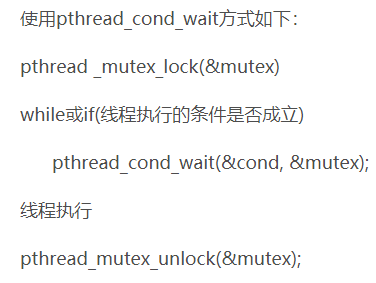
1. 管理者线程

负责管理工作线程，仅包含1个，管理工作线程使其动态变化

不停监测工作线程个数和任务个数

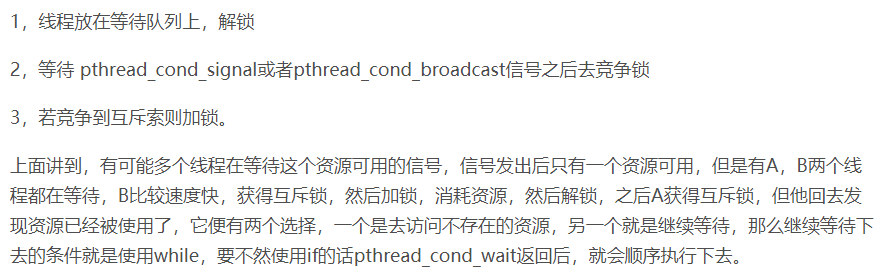
任务过多时，动态创建线程

任务过少时，动态销毁线程



线程在执行的部分访问的是进程的资源，有可能多个线程需要访问临界资源，为了避免由于线程并发引起的资源竞争，故让每个线程互斥访问共有资源。

在判断while / if条件成立时，将会调用pthread\_cond\_wait将自身阻塞（该函数的使用必须和一个互斥锁配合），并且为了防止死锁，其自身持有的互斥锁将会被释放，释放时机在其加入线程等待队列之后。若释放锁在加入等待队列以前，则某个线程可以拿到互斥锁访问共有资源从而导致原线程所等待的条件改变



线程池的工作者函数：

· 判断线程池是否为(空并且没有关闭)，若是则调用pthread\_cond\_wait阻塞消费者

· 判断是否需要销毁（销毁过程中管理者进程引导工作者进程自动销毁）

· 判断线程池是否关闭，若是则回收全部资源

· 从任务队列头部取出一个任务后调用pthread\_cond\_signal唤醒生产者

· 执行任务（任务开始结束均进入临界区改变busy\_num的值然后退出临界区）

管理者函数：每隔k秒进行一次操作

· 进入临界区取出当前忙的线程、活着的线程以及任务数量

· 根据上述数量动态添加、销毁线程

· 销毁线程是通过唤醒工作线程的方式诱导其自动退出

任务添加函数：（充当生产者）

· 判断线程池是否为(满并且没有关闭)，若是则调用pthread\_cond\_wait阻塞生产者

· 判断线程池是否关闭，若是则退出函数

· 将新任务添加到任务队列队尾

· 唤醒消费者

线程池的销毁：

· 关闭线程池，调用pthread\_join函数阻塞回收管理者线程

· 唤醒所有被阻塞的消费者线程，诱导其被销毁

· 回收空间以及临界资源和条件变量